Compte-rendu de projet technologique : Panama Papers

ANDRIANIRINA Diane
DELAR Emmanoé
SADOURI Massinissa

Avril 2018

Table des matières

1 Introduction

Le but de ce projet est de créer une base de données contenant les informations des Panama Papers avec Neo4j et les mettre facilement à disposition de l'utilisateur grâce à une architecture client/serveur Flask.

2 Gestion du projet

2.1 Outils de travail utilisés

- Trello : gestion des différentes tâches du projet (statuts), partage d'idées
- Framapad : mise en commun des différents codes Cypher

☛ Git : mises à jour du projet (codes Python Flask et HTML, fichiers CSV)

2.2 Cahier des charges et réalisation

2.2.1 Étude des besoins potentiels de l'utilisateur

Afin de visualiser facilement les données et leur donner un sens, nous avons considéré qu'il était nécessaire de répondre aux besoins suivants :

➢ Pouvoir consulter les informations concernant un pays choisi (sociétés, officiers...)

➢ Représenter les informations de manière pratique et ordonnée

➢ Pouvoir comparer le nombre de sociétés implantées entre les différents pays

2.2.2 Étape 1 : Chargement des données sur Neo4j

Besoins fonctionnels :
À partir des données contenues dans les fichiers CSV des Panama Papers, créer un noeud pour chaque donnée (entity, officer, intermediary, address) avec toutes les propriétés. Une fois les noeuds crées, les relier entre eux par les différentes relations les liant (officer of, intermediary of, registered address) qui ont été définies dans un fichier CSV spécifique (edges.csv).

Difficultés : nombre de données, de propriétés et de relations entre les noeuds très important impliquant un temps de chargement très long (en particulier pour le fichier edges.csv)

→ **Solutions :**

- indexage des données sur la propriété "id" (propre à chaque donnée) pour accélérer les filtres
- découpage du fichier edges.csv selon le type de relation pour un chargement plus rapide (3 fichiers car 3 catégories de relations)

(voir annexe)

2.2.3 Étape 2 : Modélisation interactive des données sur le site (Neo4j/Flask/HTML/D3.js)

Besoins fonctionnels :

Créer une interface pratique pour l'utilisateur afin de faciliter ses recherches et la lier à Neo4j pour obtenir des informations issues de la base de données.

➠ Modèle adapté au besoin : carte du monde interactive avec possibilité de choisir le pays désiré (représentation D3.js).

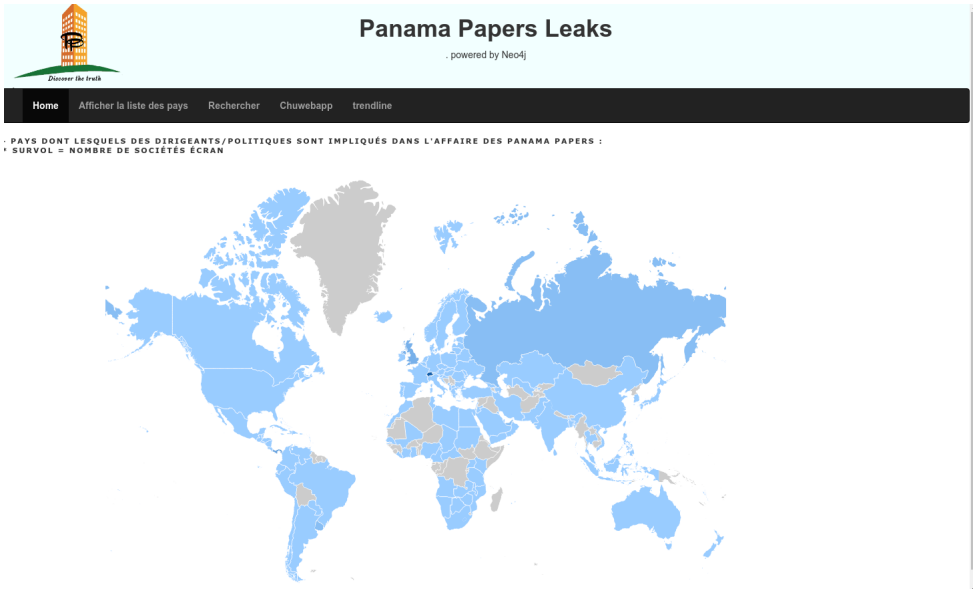

Figure 1 – Carte du monde interactive

Fonctionnement général :

La représentation graphique de la carte du monde D3.js repose sur un fichier CSV où nous avons rentré le nom de tous les pays concernés par les données, le nombre de sociétés-écran qu'ils possèdent sur leur sol (obtenu grâce à une requête à la base de données Neo4j) et leurs code pays (comme définis dans les fichiers CSV des données Panama Papers).

Premièrement, sur la carte, la couleur d'un pays est plus ou moins foncée selon le nombre de sociétés présentes. Cela permet à l'utilisateur d'avoir une vision générale de l'état des choses.

Deuxièmement, un clic sur un pays renvoie une liste-tableau organisée des noms des sociétés qui y sont, et, sur lesquels nous pouvons cliquer pour obtenir

des informations supplémentaires sur la société voulue.

Société(s) écran basée en United States

name	jurisdiction	service provider	incorporation date
XENON TRADING S.A.	NIUE	Mossack Fonseca	08-SEP-2005
MICHANNE CORPORATION	BAH	Mossack Fonseca	02-JAN-1992
FALCON TRADING & ENGINEERING CO. LTD.	NIUE	Mossack Fonseca	18-SEP-2002
WOLF'S MOUNTAIN INC.	SAM	Mossack Fonseca	19-DEC-2006
OMNI AG. S.A.	PMA	Mossack Fonseca	15-MAY-1980
HUANCA INVESTMENT LLP	UK	Mossack Fonseca	16-JUL-2015
COMMUNICATION SERVICES INTERNATIONAL INC.	PMA	Mossack Fonseca	22-NOV-1988
LODMAX PROPERTIES S.A	PMA	Mossack Fonseca	31-JUL-1989
GULLA PROPERTIES DEVELOPMENT LLC	NEV	Mossack Fonseca	25-FEB-2011
DUNTOWER EQUITIES LC	NEV	Mossack Fonseca	15-JUL-2005

FIGURE 2 – Exemple avec les États-Unis

Fonctionnement technique du renvoi d'informations :

Les données du fichier ont été lues et rentrées dans une liste grâce à une lecture CSV sur Python. Par le clic effectué sur un pays, on récupère le code-pays associé dans la liste établie précédemment. On injecte ensuite ce code-pays dans une requête Cypher pour nous renvoyer ici les "entity" (module py2neo) qui seront affichées dans un tableau dans une page HTML (template).

```python
@app.route('/vols')
def simple():
    from py2neo import Graph, DBMS, authenticate
    my_dbms = DBMS("http://localhost:7474/")
    authenticate("localhost:7474", "neo4j", "3789")
    server = "localhost:7474"
    graph = Graph(password="3789")
    sec = graph.data("MATCH (n) RETURN n.nom, n.codepays order by n.codepays") ## Augmente le temps de chargement de la route /vol
    return render_template('vols.html',titre="les vols", ville = sec)
```

FIGURE 3 – Code de requête à la base de données et affichage de la carte

☹ **Point négatif :** Chargement lent de la page vols.html (carte, route "/vols") à cause de la requête à la base de donnée Neo4j (≈ 500 mb) côté serveur (back-end).

```python

## Generate  an array with all the Countries
with open('./static/population1.csv','r') as Countrys:
    reader = csv.reader(Countrys)
    nodes = [ n for n in Countrys][2:]

country_names = csv.reader(nodes, delimiter = ',')
list_of_country = []
for country in country_names:
    list_of_country.append(country)
country_names = []; country_cpays = [];

########################################

@app.route('/<click_map>')
def test(click_map):
    for i in range(0,len(list_of_country)):
        if click_map == list_of_country[i][1]:
            print(click_map)
            cpays = list_of_country[i][6]
            print(cpays)
            nb_vol_dep = graph.data("match (e) where e.countrycodes={X} return e", X=cpays)
            return render_template('search.html', selected_country=click_map, dep=nb_vol_dep)
    abort(404) ## Message d'erreur

```

FIGURE 4 – Partie du code renvoyant les données voulues d'un pays

2.2.4 Étape 3 : Affichage pratique des données renvoyées

Besoins fonctionnels :

- Trouver une représentation (graphique ou non) adaptée à la structure des données renvoyées après avoir cliqué sur le nom d'une société-écran : relations, officers, intermediary...
- Choix pertinent des informations renvoyées à l'utilisateur

3 Prolongements

- Améliorer l'ergonomie en instaurant un système de pagination pour l'affichage d'une quantité importante de résultats
- Possibilité de rechercher directement une société par un moteur de recherche

4 Annexes

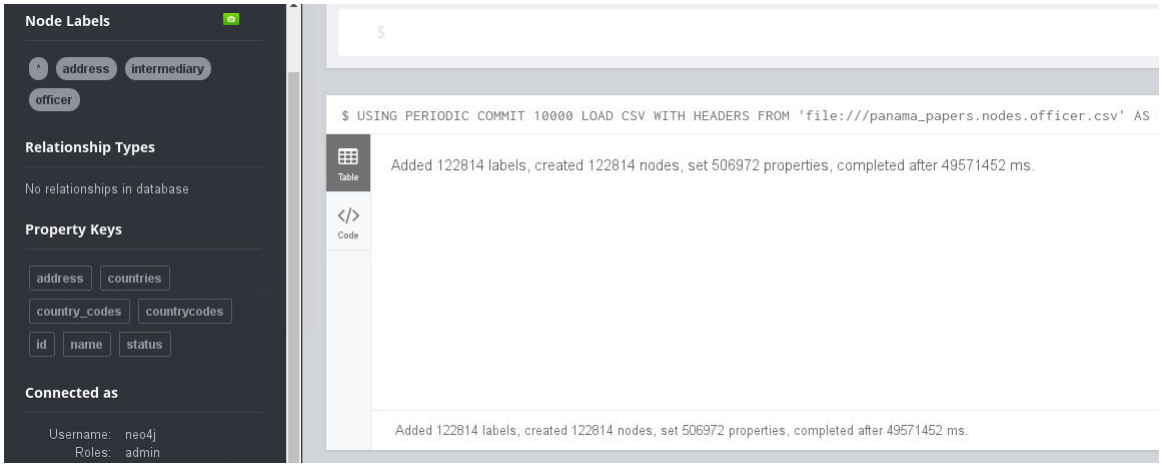

FIGURE 5 – Exemple : chargement des "officers"

```
LOAD CSV WITH HEADERS FROM 'file:///officer_of.csv' AS line
FIELDTERMINATOR ','
WITH line
match (p1:officer) where p1.id = line.`Node_1`
match (p2:entity) where p2.id = line.`Node_2`
MERGE (p1)-[:officer_of]->(p2)

LOAD CSV WITH HEADERS FROM 'file:///intermediary_of.csv' AS line
FIELDTERMINATOR ','
WITH line
match (p1:intermediary) where p1.id = line.`node_1`
match (p2:entity) where p2.id = line.`node_2`
MERGE (p1)-[:intermediary_of]->(p2)

LOAD CSV WITH HEADERS FROM 'file:///address.csv' AS line
FIELDTERMINATOR ','
WITH line
match (p1:entity) where p1.id = line.`node_1`
match (p2:address) where p2.id = line.`node_2`
MERGE (p1)-[:registered_address]->(p2)

LOAD CSV WITH HEADERS FROM 'file:///address.csv' AS line
FIELDTERMINATOR ','
WITH line
match (p1:officer) where p1.id = line.`node_1`
match (p2:address) where p2.id = line.`node_2`
MERGE (p1)-[:registered_address]->(p2)
```

FIGURE 6 - Requêtes Cypher pour la création des relations après découpage du fichier en 3 fichiers